



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/648,577	08/25/2003	Namit Jain	50277-2236	4090
<div>29989 7590 08/21/2007</div> <div>HICKMAN PALERMO TRUONG & BECKER, LLP</div> <div>2055 GATEWAY PLACE</div> <div>SUITE 550</div> <div>SAN JOSE, CA 95110</div>				
			EXAMINER	
			RADTKE, MARK A	
			ART UNIT	PAPER NUMBER
			2165	
			MAIL DATE	DELIVERY MODE
			08/21/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.



UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

MAILED

AUG 21 2007

Technology Center 2100

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Application Number: 10/648,577
Filing Date: August 25, 2003
Appellant(s): JAIN ET AL.

Christian A. Nicholes
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 2 May 2007 appealing from the Office action mailed 30 October 2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The following are the related appeals, interferences, and judicial proceedings known to the examiner which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal:

10/648,749

10/648,600

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,085,198

SKINNER et al.

7-2000

O'Reilly, "Oracle SQL*Loader: The Definitive Guide" Chapter 10, Section 10.1.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1-3 and 5-16 are rejected under 35 U.S.C. 102(b) as being anticipated by Skinner et al. (U.S. Patent 6,085,198).

As to claim 1, Skinner et al. teaches a method of storing data into a database (see Abstract), the method comprising:

a loader application receiving data (see figure 3, Comm Mgmt 305B and figure 4, step 400 and column 16, lines 48-49);

determining one or more routines that are associated with a type of said data, wherein said one or more routines are implemented by a program that is external to both said loader application and a database server that manages said database (see column 16, lines 49-55, where "routines" is read on "methods");

invoking said one or more routines (see column 18, lines 6-10);

in response to said one or more routines being invoked, said program performing steps comprising:

creating a data structure that has one or more elements that correspond to one or more attributes of said type (see column 16, lines 60-62 and figure 4, step 404); and

populating said one or more elements with one or more values that are specified in said data, wherein said one or more values correspond to said one or more attributes (see column 30, lines 60-67);

generating, based on said data structure, a data stream that conforms to a format of data blocks of said database (see column 31, lines 1-2); and

writing said data into one or more data blocks in said database (see column 31, lines 23-33).

Art Unit: 2165

As to claim 2, Skinner et al. teaches wherein a number of attributes of said type is not defined to said loader application (See column 17, line 65 – column 18, line 5. Attributes can be determined by calling functions instead of loading documents).

As to claim 3, Skinner et al. teaches wherein a type of an attribute of said type of said data is not defined to said loader application (See column 17, line 65 – column 18, line 5. Attributes can be determined by calling functions instead of loading documents).

As to claim 5, Skinner et al. teaches wherein said data structure is created by said program in a memory space of said loader application rather than a memory space of said program (see column 2, lines 45-47, "client tier and application tier which permit instantiation of the generated data classes").

As to claim 6, Skinner et al. teaches wherein said determining comprises locating addresses of one or more routines that are in a same entry of a table as an identity of said type (see column 16, line 40, "associated data types").

As to claim 7, Skinner et al. teaches further comprising:
adding, to a table, an entry that indicates an association between said type and said one or more routines (see column 19, lines 66-67 and column 20 lines 15-19).

Art Unit: 2165

As to claim 8, Skinner et al. teaches wherein said invoking comprises invoking one or more routines that are located at one or more addresses that are associated with said type via an associative structure (see column 18, lines 6-10).

As to claims 9-11, 13-16 and 19-20, Skinner et al. teaches a computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the methods recited in claims 1-8 (see column 5, lines 50-57).

As to claim 17, Skinner et al. teaches further comprising:

said program registering, with said loader application, said one or more routines, which are not implemented by said loader application (See columns 32-35, "Reflection Methods" and see also column 24, lines 12-45. Routines not implemented by the loader application are routines of opaque types, as defined in paragraph [0021] of the instant specification. Reflection methods allow the discovery of the internal structure of objects, even when the structure is hidden by encapsulation. As defined in the specification, and as used in the claims, opaque types are indistinguishable from JavaBeans or other objects defined by an interface. Skinner et al. teaches this registration and discovery for any type of object. See "Using Java Reflection" and "Java programming dynamics, Part 2: Introducing reflection" for well-known uses of reflection to "look into" objects.); and

in response to said program registering said one or more routines with said loader application, said loader application adding, to a dispatch table, an entry that

Art Unit: 2165

indicates an association between said one or more routines and an opaque type implemented by said program (see Examiner's comments regarding claims 7-8).

As to claim 18, Skinner et al. teaches wherein invoking said one or more routines comprises:

said loader application invoking at least one of said one or more routines to find out (a) a number of one or more attributes within an opaque type and (b) one or more types of said one of more attributes within said opaque types (see column 19, lines 14-17 and column 20, line 50 – column 21, line 19); and

said loader application invoking at least one of said one or more routines to populate, with values of instances of the opaque type, elements of an array that is stored in a memory space of said loader application (see column 18, lines 30-40 and see also column 19, lines 39-50, "MetaSchema").

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Art Unit: 2165

4. Claim 4 is rejected under 35 U.S.C. 103(a) as being unpatentable over Skinner et al. as applied to claim 1 above, and further in view of O'Reilly (Chapter 10, section 10.1 of "Oracle SQL*Loader: The Definitive Guide").

As to claim 4, Skinner et al. teaches wherein said creating, said populating, said generating, and said writing are performed without causing a Structured Query Language (SQL) engine to load said data (see column 18, lines 8-12 where "without causing a SQL engine to load said data" is read on "extracted and loaded directly").

Skinner et al. does not explicitly teach using a direct path loading approach rather than a conventional path loading approach.

O'Reilly teaches using a direct path loading approach rather than a conventional path loading approach (see Chapter 10, Title).

Therefore, it would have been obvious to one of ordinary skill in the relevant art at the time the invention was made to have modified Skinner et al. by the teaching of O'Reilly because "the schema metadata may be [...] loaded directly" (Skinner et al., column 18, lines 9-10).

As to claim 12, Skinner et al., as modified, teaches a computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the methods recited in claims 1-8 (see column 5, lines 50-57).

(10) Response to Argument

In response to Appellant's arguments that "Skinner does not disclose, teach, or suggest that these database tables are created by a program that is external to a database server or in response to the invocation of routines that are implemented by such an external program", the arguments have been fully considered but are not deemed persuasive.

Columns 3 through 13 of Skinner describe the operation of the well-known "three-tier framework". In a three-tier framework, data propagates from a database to a client for display and modification. Changes or additions to the database are sent from the client, serialized, and finally stored in the database. Figure 4 and the accompanying portions of the specification (beginning at column 16, line 45) describe the database table creation process. As stated in the Advisory Action of 26 February 2007, a database server may be interpreted as a physical server (hardware) or a logical program (software). If the database server were considered to be a software server application (such as an Oracle database), then the program of Skinner would be a distinct software program, and thus "external" (logically) to the loader application and the database server. The program of Skinner invokes the creation of the database tables (see, for example, column 16, lines 60-62), therefore it creates the tables.

In response to Appellant's arguments that "Skinner does not disclose, teach, or suggest that these data class definitions are created by these tiers", the arguments have been fully considered but are not deemed persuasive.

As described at column 15, line 7 – column 16, line 12, data updates (which may include the creation of additional classes as specified at column 15, lines 60-63) propagate from the client tier to the database ("the transaction begins in application logic and GUI component 301A", which resides in the client tier in figure 9). Thus, the client tier creates the class definitions because it causes the database to create the class definitions. As in the previous argument, Appellant fails to consider that the non-database tiers of the application invoke the creation indirectly. Such an interpretation is overly narrow and ignores the common jargon used by those skilled in the art. It would be similar to arguing that it is impossible to "create a data structure" because matter cannot be created or destroyed.

In response to Appellant's arguments that Skinner does not teach "generating, based on said data structure, a data stream that conforms to a format of data blocks of said database", the arguments have been fully considered but are not deemed persuasive.

Skinner writes data to the database. The last two limitations of claim 1 are broad and merely require writing data to a database. This is clearly taught by Skinner. A database without data written to it (or with data written to it in a different format) would be non-functional. It is noted that the claims are not indented under the "in response to" step, therefore it is assumed that they may be carried out by any part of the system.

In response to Appellant's arguments that Skinner does not teach "adding, to a table, an entry that indicates an association between said type and said one or more routines", the arguments have been fully considered but are not deemed persuasive.

In an object-oriented environment, all objects of a class share the same routines but may have different data stored in the data members. Skinner consolidates these shared routines in a class description and then attaches them from this common location when the Factory pattern builds objects. See figure 6B, step 613 and see columns 28-29, "Specified Methods". Specifically, "those methods described by the MetaMethod elements of the Vectors 'myMethods' [...] are added to the common interface" (column 28, lines 41-42). Also, in lines 28-30 of column 21, Skinner et al. discloses, "A reference, 'myPassToAttribute,' refers to the MetaAttribute instance describing an attribute to forward the method call to, if applicable." A reference is an address. Thus, Skinner et al. teaches locating (which necessarily requires "associating") addresses of methods for a given type.

In response to Appellant's arguments that Skinner does not teach that "such 'data class definitions' are created by a program that implements the methods described by the 'MetaMethod' instances in response to the invocation of those methods", the arguments have been fully considered but are not deemed persuasive.

Data class definitions can be created on-demand when a request is sent by the client. See column 4, lines 10-19. "Class factories are also generated that provide factory methods for creating new instances of the data classes." See also column 12,

Art Unit: 2165

lines 44-62. "If the data object is not in server-side object cache component 303B, a new data object is obtained, via factory method calls directed to a data class factory object, to encapsulate the data." This is done in response to "receiv[ing] query objects from the client tier" (column 12, line 33). Thus the client invokes the construction of objects just as it invokes the construction of classes and tables. If A causes B which causes C, then A is the cause of C. Similarly, the client causes all of the claimed behaviors to be carried out on the server, so the client is the cause of the behaviors.

In response to Appellant's arguments that Skinner does not teach "wherein said invoking comprises invoking one or more routines that are located at one or more addresses that are associated with said type via an associative structure", the arguments have been fully considered but are not deemed persuasive.

A database table is an associative structure. A Vector of references is an associative structure.

In response to Appellant's arguments that Skinner does not teach "that the routines are of or within or part of any opaque type" or "'said loader application invoking at least one of said one or more routines to find out (a) a number of one or more attributes within an opaque type and (b) one or more types of said one or more attributes within said opaque type", the arguments have been fully considered but are not deemed persuasive.

See columns 32-35, "Reflection Methods" and see also column 24, lines 12-45. Routines not implemented by the loader application are routines of opaque types, as defined in paragraph [0021] of the instant specification. Reflection methods allow the discovery of the internal structure of objects, even when the structure is hidden by encapsulation. As defined in the specification, and as used in the claims, opaque types are indistinguishable from JavaBeans or other objects defined by an interface. Skinner et al. teaches this registration and discovery for any type of object. See "Using Java Reflection" and "Java programming dynamics, Part 2: Introducing reflection" for well-known uses of reflection to "look into" objects.

In response to Appellant's argument that the rejection of claims 4 and 12 is based entirely on hindsight reasoning, the arguments have been fully considered but are not deemed persuasive.

As argued above, Skinner teaches the serialization and deserialization of opaque types. O'Reilly is cited for teaching direct path loading. Direct path loading is a well-known technique that would have yielded predictable results to one of ordinary skill in the art at the time of the invention. There are no facts that would lead one to expect that opaque types, properly serialized by Skinner, could not be processed for direct path loading.

Art Unit: 2165

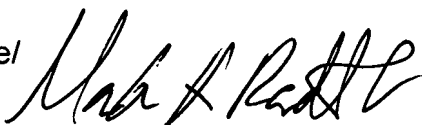
(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Mark A. Radtke/



Mark A. Radtke

Appeal Conference held on 14 August 2007. Agreement was reached to proceed to the Board of Appeals and Interferences.



Jeffrey Gaffin

JEFFREY GAFFIN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100



Hosain Alam

HOSAIN ALAM
SUPERVISORY PATENT EXAMINER